IT620: Object Oriented Programming

Instructor:	Sourish Dasgupta
Prerequisites:	Programming in C++ or Java
Slot:	M.Sc. IT Semester-II
Category:	Core
Course Credits(L--T--P--Cr):	3--0--2--4
Lectures:	Yes (Offline)
Lab and Practical:	Yes
TA contact info:	TBD

Course Description:

This course provides an in-depth exploration of Object-Oriented Programming (OOP) with a strong emphasis on design patterns, equipping students with the skills to develop robust, maintainable, and scalable software systems.

Course Objectives:

- **Understand Core OOP Principles:** Gain a solid foundation in OOP concepts, including encapsulation, inheritance, polymorphism, and abstraction.
- **Apply Design Patterns:** Learn to recognize and implement common design patterns to address recurring software design challenges effectively.
- **Develop Maintainable Code:** Employ best practices to write modular and reusable code, enhancing software maintainability and scalability.

Course Structure

- **Lecture:** Learn the foundational concepts of modern industry-standard Object Oriented Design
- **Project:** The course will be project-driven, where a specific application-oriented problem will be defined and given. Every lecture will be designed in the context of solving the given problem with an introduction to necessary technologies. Bi-weekly assignments will be given, and assignments will be designed as necessary stepping stones toward the completion of the project.

Suggested Books:

- "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.
- "Head First Design Patterns" by Eric Freeman and Elisabeth Robson.
- Online courses and tutorials on OOP and design patterns.

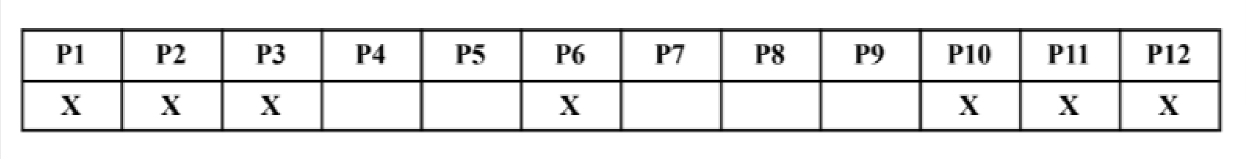
Course Outcomes:

By the end of this course, students will be able to:

- Apply OOP principles to design and implement software solutions.
- Identify and utilize appropriate design patterns to solve common design problems.

IT620: Object Oriented Programming

- Develop code that is modular, reusable, and easy to maintain.
- Critically analyze and improve existing codebases using design patterns.



P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
X	X	X			X				X	X	X

Evaluation Scheme

- **Mid-semester Exam:** 20 %
- **End-semester Exam:** 30 %
- **Group Project-Assignments:** 50 % (group size will be a maximum of 4 students)

Grading Policy

For Credit:

AA: >=85%; AB: >=75%; BB: >=65%; BC: >=55%; CC: >=45%; CD: >=35%; DD: >=25%; F: <25%

For Audit:

Pass: >=25%

Course Plan:

Units	Topics	Number of Lectures
Introduction to Object-Oriented Programming	● Fundamental concepts and principles of OOP ● Benefits of using OOP in software development	2
Core OOP Concepts	● Encapsulation: Protecting object integrity by restricting access to internal states ● Inheritance: Creating hierarchical relationships between classes ● Polymorphism: Designing objects to share behaviors, allowing for flexible code.	8

IT620: Object Oriented Programming

	● Abstraction: Simplifying complex systems by modeling classes appropriate to the problem.	
Introduction to Design Patterns	● Definition and significance of design patterns in software engineering. ● Overview of the "Gang of Four" design patterns.	4
Creational Patterns	● Factory Method, Abstract Factory, Singleton, Builder, Prototype. ● Techniques for object creation to enhance flexibility and reuse.	4
Structural Patterns	● Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy. ● Organizing classes and objects to form larger structures and interfaces.	5
Behavioral Patterns	● Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor. ● Managing object collaboration and responsibility distribution	7
Applying Design Patterns in OOP	● Identifying appropriate patterns for specific scenarios. ● Implementing patterns in various programming languages. ● Evaluating the impact of design patterns on code quality and maintenance.	3
Advanced Topics	● Anti-patterns: Recognizing and avoiding common design pitfalls. ● Integration of design patterns with modern development practices.	3

Lectures: 36 (tentative)